
IDUG
Session: K01
Best Practices for IMS Database Reorganization
Janet LeBlanc
IBM Corporation
IDUG 2008
North America
Experience IDUG
May 19, 2008 • 09:45 a.m. – 10:45 a.m.
Platform: IMS for z/OS

Session K01

Best Practices for IMS Database Reorganization

Janet LeBlanc

IBM Corporation

leblancj@ca.ibm.com

Agenda

- Why Reorganization?
- Performance
- Reorganization Criteria
- Reducing Reorganization Frequency
- Reorganization Performance Opportunities

Are Reorganizations a Luxury or a Necessity?

What is the purpose of Reorganizations

- What is it?
  - Process of changing the physical storage and/or structure of a database to better achieve the application's performance requirements

Two types of Reorganization

- Physical Reorganization
  - Optimize the physical storage of the database
- Re-structure Reorganization
  - Alter the structure of the database

Reasons for Physical Reorganization

- To reclaim and consolidate free space that has become fragmented due to repeated insertion and deletion of segments.
- To optimize the physical storage of the database segments for maximum performance
  - get dependent segments that are in distant blocks, increasing physical I/O, back in the same block as the parent and/or root).
  - this situation is normally the result of high update activity on the database

What stops us from doing Reorganizations??

- Perceived Costs
  - People time
  - Computer costs
- No one is complaining about performance so reorganization must not be needed
- Availability
  - 24 X 7 – no time to do it

But what is the REAL Reason for Reorganization??

APPLICATION PERFORMANCE!!!

Performance

Definition - A performance problem is generally noted as bad or erratic response times or an unacceptable amount of resource usage

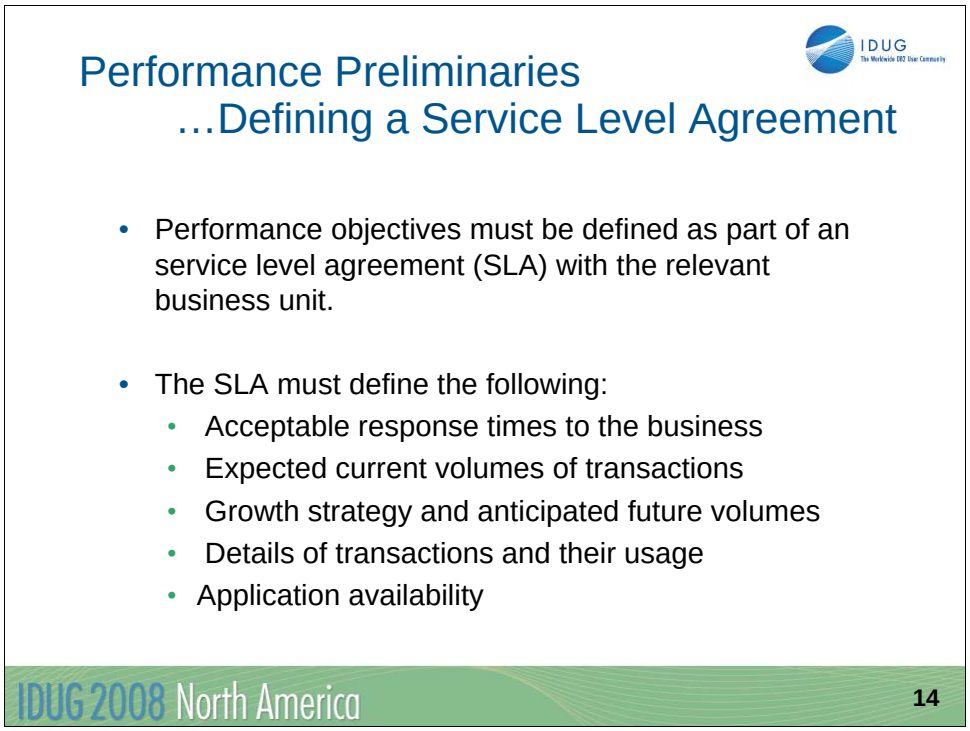
What causes us to investigate Performance

- Service level objectives not being met
- Users complaining about slow response
- Unexpected changes in response times or resource utilizations
- z/OS operating system showing signs of stress
- The throughput on the system is erratic
- Changes in workload which were not anticipated
- Changes in the profile of transactions

How to justify

- Where are my performance opportunities
  - What is the potential improvement
    - CPU time
    - Elapsed time
    - EXCPs
    - Processes not needed
- Track before/after statistics to prove value of reorganization and performance tuning
- Proof of Concept of reorganization to justify time and expense

Application Performance



It must not come as a surprise that we need the ability to track and trend transactions across

all business units throughout the organization. This tracking and trending requires the use of

sophisticated tools and a repository to manage each transaction with its profile. Depending on

what we have agreed to in our SLA, we might need to collect, summarize, and store statistics

Definition: A performance problem is generally noted as bad or erratic response times or

an unacceptable amount of resource usage.

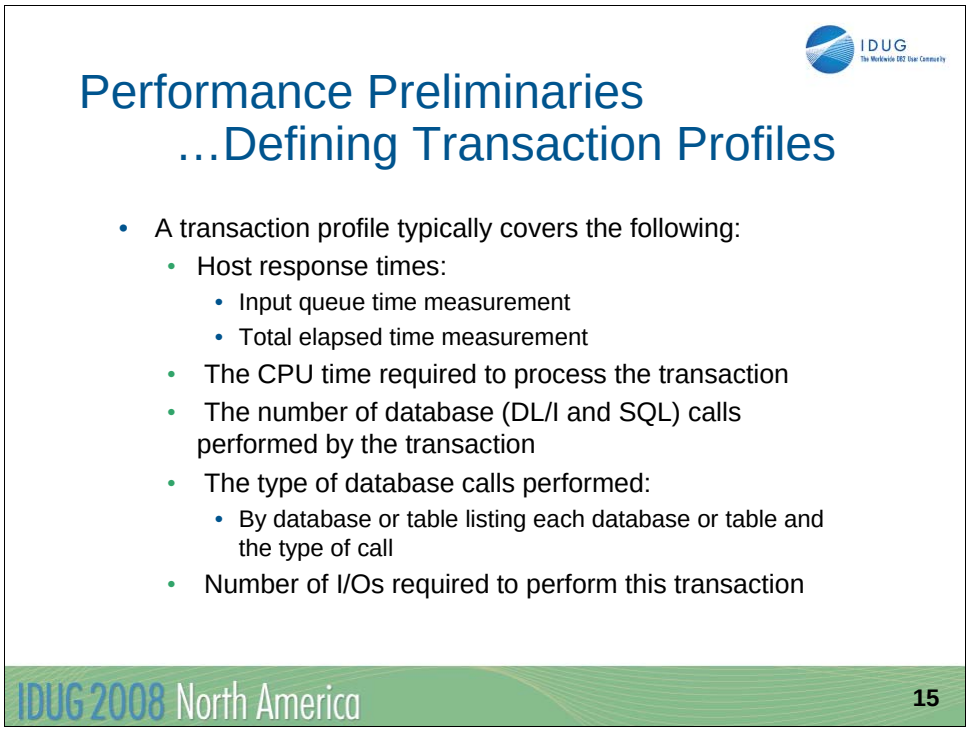
Chapter 1. Defining the performance problem in an IMS environment **3**

and profiles on all transactions 24 hours a day, 365 days a year in a repository. There are

many ways to create a repository and many products that provide you with a view to access

this repository. These tools must have the capability to perform statistical analysis of the data

stored in the repository.



A typical transaction generally performs many different types of functions. A transaction

designated to process internet banking can perform transfers and payments as well as

balance inquiries. The profile by transaction is different for each of these functions. Analysis

must be done to quantify these profiles to percentiles of the various functions of a transaction

in order to make meaningful decisions.

This implies that making changes to a very low percentile-used function would not be

beneficial, when compared to the total amount of CPU consumed and DL/I calls made. Cost

as opposed to effort becomes a factor in most decisions.

When you have a transaction profiling repository with daily feeds into the repository from the

IMS log records, other performance log records generated by vendor products, and SMF

records, analysis and interpretation of the data become crucial issues.

A typical transaction generally performs many different types of functions. A transaction

designated to process internet banking can perform transfers and payments as well as

balance inquiries. The profile by transaction is different for each of these functions. Analysis

must be done to quantify these profiles to percentiles of the various functions of a transaction

in order to make meaningful decisions.

This implies that making changes to a very low percentile-used function would not be

beneficial, when compared to the total amount of CPU consumed and DL/I calls made. Cost

as opposed to effort becomes a factor in most decisions.

When you have a transaction profiling repository with daily feeds into the repository from the

IMS log records, other performance log records generated by vendor products, and SMF

records, analysis and interpretation of the data become crucial issues.

What do I need?

- Application Details
- Baseline Statistics & Historical Statistics
  - Baseline can be different things depending on what you are trying to do
    - Peak load
    - Quiet times
    - Above just after a database reorg
    - Before/After Performance Tuning
- Current Statistics
- Database Definition

Performance Reporting

- Daily monitoring
  - Transit response time reports
  - Management exception report
- Performance problem? Look into the details!
  - Bad response time? Transit reports
  - IMS resource constraint? Resource Utilization reports
- Long-term capacity planning and service levels
  - Transaction History File – daily transaction performance
  - Load into DB2 to build a Performance Database
  - Report on host or workstation using your favorite SQL reporting tool

IMS V10 - The 56FA log record

- One record per transaction rather than per schedule (type 07)
- Additional information including:
  - OSAM and VSAM read and write counts
  - Database IO counts and elapsed times
  - Database lock elapsed times
  - External subsystem call counts
  - UOR elapsed and CPU times

Summary report – transaction activity statistically summarized

IMS Performance Analyzer
Transaction Transit Summary

SUMM0001 Printed at 15:16:04 07Feb2007
Data from 09.16.38 07Feb2007 to 09.43.39 07Feb2007

Trancode	Tran Count	Avg InputQ Time	Avg Process Time	Avg CPU Time	Avg Total IO Count	Avg DB IO Time	Avg DB Lock Time
IVTNO	52	521.346	295.452	1.158	0	2.056	0.000
IVTNV	36	254.697	685.690	1.555	237	217.394	0.000
Total	116	516.315	302.815	1.165	5	6.119	0.000

V10 allows microsecond precision

Avg OSAMRead Count	Avg OSAMWrit Count	Avg VSAMRead Count	Avg VSAMWrit Count	Avg ESAFcall Count
0	0	0	0	6
0	0	3	234	8
0	0	0	4	7

Form-based reporting

- **Summarize transaction activity based on any criteria, for example Region Type**
- **Statistical functions include average and peak percentile (to measure SLA adherence)**

Transaction Dashboard

DASH Printed at 14:34:54 05May2006 Data from 16.03.39 29Dec2005 to 16.17.33 29Dec2005

Reg Typ	Tran Count	Avg InputQ Time	Avg Process Time	Avg OutputQ Time	Avg Total Time	Avg IMS Resp Time	Avg CPU Time	90% InputQ Time	90% Process Time	90% OutputQ Time	90% Total Time	90% IMS Resp Time	90% CPU Time
BMP	287	200	409	0	607	0	0	819	1282	0	1543	0	0
DBC	1	0	5	0	5	0	0	0	5	0	5	0	0
MPP	47017	118	63	8	189	183	18	3550	982	26	3827	3908	227
MSC	204	0	97	14	111	111	0	0	267	35	280	280	0

Summarization by Region Type

Transit time breakdown (averages)

Transit time breakdown (90% SLA)

Transaction volume for the day

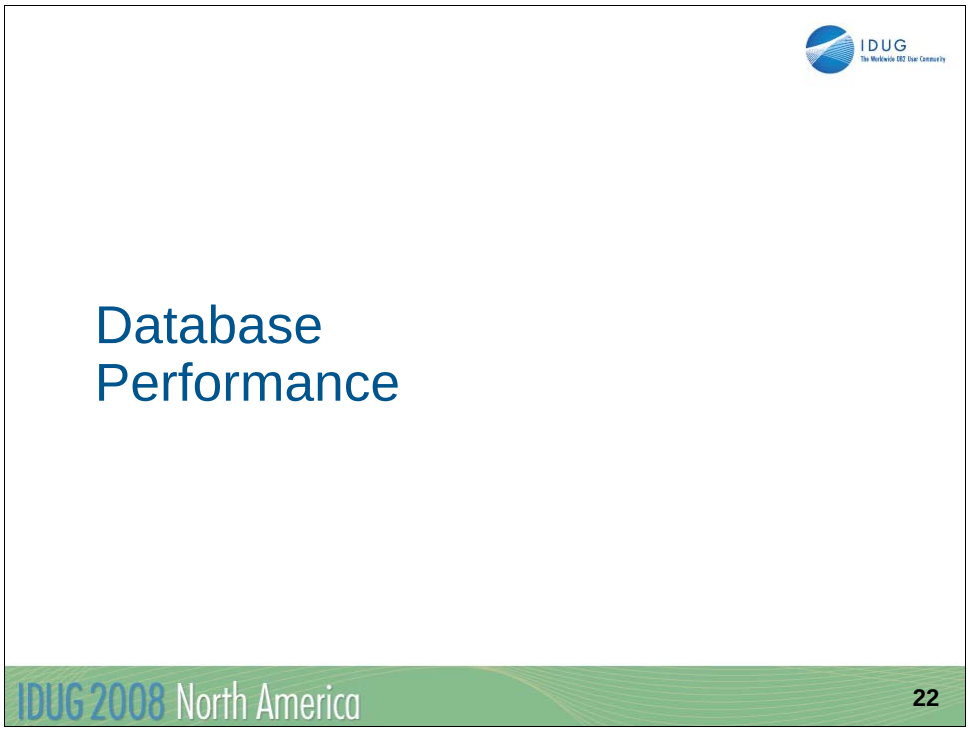
CPU time

Design a report to meet your needs

Instead of the usual summarization by transaction code name, tweak the form to summarize by “Region Type”.

Now the report provides a high-level overview of system activity based on the types of transactions that run in your environment.

Peak percentiles can detect if transactions are performing at the required level (Service Level Agreement)



Database/Application Efficiency Indicators: The two prime indicators of database and application efficiency appear below **DATABASE CALLS** in the **Calls/Tran** and **IWTs/Call** columns. Database and application design objectives should minimize the number of DL/I calls per transaction and the number of IWAITs per DL/I call. Inefficient database design leads to both an excessive number of I/Os per call and contention between applications for database records. Applications should be sensitive to their influence on response time and avoid issuing redundant or inefficient call patterns. A high number of DL/I calls per transaction increases response time and indicate inappropriate call patterns for a response-time sensitive environment. Either the application should be a BMP and scheduled as such, or the application should be examined as a candidate for redesign. See the Program Chapter 5. Analyzing Monitor reports **285**

Summary report to identify applications that issue many DL/I calls; then see the PSB Details or PSB-Transaction Code Analysis reports for CALL activity for specific databases. A high number of IWAITs per DL/I call also increases response time and indicates several possibilities. First, whatever the acceptable guideline for an installation, IWAITs per call should remain fairly constant with time. This value should be monitored regularly to detect an increasing rate of IWAITs per call, which indicates that: v Twin chains are developing, signalling the need for reorganization v The HDAM randomizer algorithm may no longer be effective DBT can be useful in analyzing these conditions. A high number of IWAITs per call may also indicate heavy use of logical relationships and/or secondary indexing. Review the PSB Details-PCB Totals report to assess these possibilities. Applications may be responsible for the IWAITs per call rate based on their DL/I call patterns. Look at the Program Summary report for PSBs with high IWAITs per call and then the PSB-Transaction Code Analysis and/or Program Trace reports. Significant performance improvements have been realized through application redesign that eliminates unnecessarily complex or redundant calls. The report on page 275 shows 0.32 IWAITs per call, which is below the guideline figure of 0.5. The Database IWAIT Summary can be examined to identify specific violations of the low IWAITs per call objectives. However, the most convenient way to monitor IWAITs per call violations is to set the maximum value for the IWAITs per call distributions to the installation performance objective; then the Exception Listing report can pinpoint the violations. Refer to the PSB Details reports to examine IWAITs per call data for each PCB/PSB combination.

Database IWAIT Analysis

Report from 08Jun2006 13.06.12.71 IMS 8.1.0 IMS Performance Analyzer 4.1 Report to 08Jun2006 13.10.39.26

Database IWAIT Analysis

Region Totals From 08Jun2006 13.06.21.06 To 08Jun2006 13.09.52.04 Elapsed= 0 Hrs 4 Mins 26.545.110 Secs

DDname	Type	IWAITs	Elap/IWAIT Sc.Mil.Mic	StdDev % Avg	Max IWAIT Sc.Mil.Mic	Calls Waiting	IWAITs /Call	Pct Tot Calls	Pct Tot IWAITs	Pct Tot IWTElp	Pct Tot DLAElp
DB23AR0	DEDB	5	3.517	.707	5.827	5	1.00	.68%	3.82%	4.955%	.059%
DB23AR1	DEDB	12	4.263	1.302	12.751	9	1.33	1.23%	9.16%	14.414%	.172%
DB23AR2	DEDB	34	9.631	3.471	9.402	34	1.00	4.64%	25.95%	6.042%	.072%
DB23AR3	DEDB	16	1.652	2.507	12.726	16	1.00	2.19%	12.21%	7.449%	.089%
DB23AR4	DEDB	3	10.386	.701	19.950	3	1.00	.41%	2.29%	8.779%	.105%
DB23AR5	DEDB	31	2.635	1.754	11.386	21	1.48	2.87%	23.66%	23.016%	.275%
DD01AR0	DEDB	28	3.958	1.230	14.039	21	1.33	2.87%	21.37%	31.231%	.373%
DIMSQ1D1	VSAM	1	14.541	.000	14.541	1	1.00	.14%	.76%	4.097%	.049%
SHMSG	QUE	1	0.055	.000	0.055	1	1.00	.14%	.76%	.015%	.000%
**Grand*	*Tot	131	2.709	1.761	19.950	111	1.18	15.16%	100.00%	100.00%	1.194%
DEDB	*Grp	129	2.638	1.776	19.950	109	1.18	14.09%	98.47%	95.887%	1.144%
VSAM	*Grp	1	14.541	.000	14.541	1	1.00	.14%	.76%	4.097%	.049%

Database Update Activity

Start 14Jun2006 10.15.00.00 — IMS Performance Analyzer — End 14Jun2006 10.16.00.00 PAGE 1

Data Base Update Activity-IMSW

Database	DDname	Blocks Updated	** Generated Update Counts ** Inserts	Deletes	Replaces	DB Open Calls	****First Update**** Date	Time	*****Last Update**** Date	Time
DATTENT		2	0	0	0	0	14Jun2006	10.15.03.90	14Jun2006	10.15.39.40
DBURAUX	DBURAUX	0	0	0	0	0				
DCOMPTE		2	2	0	5	0	14Jun2006	10.15.03.90	14Jun2006	10.15.19.80
		2	4	0	0	0	14Jun2006	10.15.03.90	14Jun2006	10.15.19.80
DECHTXT		32	64	0	17	0	14Jun2006	10.15.00.50	14Jun2006	10.15.56.90
DHISTOR		1	5	0	0	0	14Jun2006	10.15.03.70	14Jun2006	10.15.03.70
		1	1	0	0	0	14Jun2006	10.15.03.70	14Jun2006	10.15.03.70
DIDXATT		27	0	2	0	0	14Jun2006	10.15.03.90	14Jun2006	10.15.39.40
DIDXCON		17	1	1	0	0	14Jun2006	10.15.03.70	14Jun2006	10.15.03.70
DIDXNOM		17	0	0	2	0	14Jun2006	10.15.39.40	14Jun2006	10.15.39.40
DMATQSD	DMATQSD	0	0	0	0	0				
DMATQSI	DMATQSI	0	0	0	0	0				
DMEMOIR		61	82	0	14	0	14Jun2006	10.15.01.20	14Jun2006	10.15.59.70
DRECRSS	DRECRSS	0	0	0	0	0				
DRPETAT	DRPETAT	0	0	0	0	0				
	DRPETAT2	0	0	0	0	0				
DRPJOB	DRPJOB	0	0	0	0	0				
	DRPJOB2	0	0	0	0	0				
DSAISIE		47	3	0	1	0	14Jun2006	10.15.07.40	14Jun2006	10.15.39.70
DSOCIET		7	3	2	21	0	14Jun2006	10.15.03.70	14Jun2006	10.15.41.10
		1	1	0	0	0	14Jun2006	10.15.03.70	14Jun2006	10.15.03.70
DSOCTXT		2	4	0	0	0	14Jun2006	10.15.39.10	14Jun2006	10.15.39.20
DSTBUR		27	1	0	1	0	14Jun2006	10.15.02.90	14Jun2006	10.15.02.90
Total		121	171	13	61	0				

Database Updates by Program

Start 12Jul2006 05.47.12.73 — IMS Performance Analyzer — End 12Jul2006 11.02.17.78 Page 1

Database Update Activity-IMS

Database	Program	DDname	Blocks Updated	** Generated Update Counts ** Inserts	Deletes	Replaces	DB Open Calls	****First Update**** Date	Time	*****Last Update***** Date	Time
QDBINDXD	KDSCPDD	QDBINDXD	0	0	0	0	1				
QDBINDXX	KDSCPDD	QDBINDXX	0	0	0	0	1				
QDINDEXD	KDSCPDD	QDINDEXD	0	0	0	0	1				
QDINDEXX	KDSCPDD	QDINDEXX	0	0	0	0	1				
QESUEOKD	KDSCPDD	QESUEOKD	0	0	0	0	1				
QGJIGTXD	KDSCPHH	QGJIGTXD	48	141	21	0	1	12Jul2006	08.55.19.68	12Jul2006	10.57.59.93
QGJIGTXX	KDSCPHH	QGJIGTXX	69?	48	21	0	1	12Jul2006	08.55.19.68	12Jul2006	10.57.59.93
QGKAITRD	KDSCPFF		136	372	0	0	0	12Jul2006	09.03.20.45	12Jul2006	10.48.54.86
	KDSCPHH	QGKAITRD	33	84	0	0	1	12Jul2006	08.55.17.91	12Jul2006	10.57.59.93
QGKAITRX	KDSCPFF		128?	134	0	0	0	12Jul2006	09.03.20.45	12Jul2006	10.48.54.86
	KDSCPHH	QGKAITRX	29?	29	0	0	1	12Jul2006	08.55.17.91	12Jul2006	10.57.59.93
QGKAKEID	KDSCPFF		4	0	0	1	0	12Jul2006	09.44.36.75	12Jul2006	10.28.38.72
	KDSCPHH	QGKAKEID	28	0	0	28	1	12Jul2006	08.55.17.91	12Jul2006	10.57.59.93

When to Reorganize

Classic Reasons:

1. Extents
2. Freespace Statistics
3. CI/CA Splits
4. HDAM - % of roots in home block.

- But does that tell the true story:
  - Is the area of the database that is “out of order” accessed by applications?
  - Is it affecting application performance?
  - Gather stats post-reorg so you can tell if a reorg did make a difference
    - Application performance deteriorating
    - Too many physical I/Os to DASD
      - Check buffers TOO!

IMS HP Pointer Checker

Space Exception Reporting

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS - SPMN          "SPACE MONITOR EXCEPTION REPORT"                    PAGE:     1
5655-K53                                                  DATE: 07/11/2006  TIME: 15.40.18                  FABKSPMN - V2.R2

 MEMBER NAME : N/A
 ------------------

 DBNAME         DDNAME      DSNAME                                          DBORG ACCM     CISP  CASP  UNIT    REORGDATE    HDPCDATE
 TYP    PRI    SEC EXT AEXT      ALLOC %FSP %NRUS    TOTBLK BLKSZ LRECL  MKSEG ACTMK      ROOTS   TOTALSEG VOLSER EXT     ALLOC %USE
        DATASET SIZE    %SZ
 --- ------ ------ --- --- --------- ---- ----- --------- ----- ----- ------ -----  ---------  --------- ------ --- --------- ----

 HISAMDB1       HISAMDS1    TESTDS.PUBLIC.SAMPLE.HISAMDS1                    HISAM KS-U         0      0  3390-3 07/06/2006  07/10/2006
 CYL     50     50  1  118         50   98  N/A          98  8192   510   N/A    N/A          130         584 SYS004  1         50    2

           737,280    0.0
 *****  MORE THAN  1 EXTENTS  *****

 HISAMDB1       HISAMDS2    TESTDS.PUBLIC.SAMPLE.HISAMDS2                    HISAM ES-U       N/A   N/A  3390-3 07/06/2006* 07/10/2006*
 CYL     50     20  1  118         50   68* N/A*       1419  8192   512   N/A    N/A            0      106017 SYS004  1         50   32

          11,624,448    0.3
 *****  LAST HDPC RUN OLDER THAN  0 DAYS  *****

 TPFOH1         TPFOH1AA    TESTDS.PUBLIC.SAMPLE.TPFOH1.A00001               PHDAM ES-U       N/A   N/A  3390-3 07/06/2006  07/10/2006
 CYL     50     50  1  118         50   47    7       19045   512   505   246    246       11Q00       00037 SYS004  1         50   54
 *****  USED SPACE EXCEEDS   0 %  *****

          10,169,640    0.2
 *****  LESS THAN  50 % FREE SPACE  *****
 *****  DATASET SIZE %   0.2 EXCEEDS   0 %  *****
 *****  DATASET SIZE EXCEEDS    1 M  *****

 TPFOH2         TPFOH2AA    TESTDS.PUBLIC.SAMPLE.TPFOH2.A00001              PHIDAM ES-U       N/A   N/A  3390-3 07/06/2006  07/10/2006
 CYL     50     50  1  118         50   89    7        4410   512   505   122    122        9Q00        10178 SYS02D  1         50   12

          2,257,920    0.1
 *****  LAST REORGANIZATION DATE IS MORE THAN   2 DAYS BEFORE  *****

 TPFOX1         TPFOX1AA    TESTDS.PUBLIC.SAMPLE.TPFOX1.A00001               PSINDX KS-U         0      0  3390-3 07/06/2006  07/10/2006
 CYL     10     10  1  118         10   87  N/A          1470   512    54   N/A    N/A          N/A         9178 SYS02F  1         10   20

           752,640    0.0
 *****  CA SPLITS %   0 EXCEEDS   0 % , TOTAL CA #         2  *****
 *****  CI SPLITS %   0 EXCEEDS   0 % , TOTAL CI #      1470  *****
```

% Roots in Home Block

```
IMS HIGH PERFORMANCE POINTER CHECKER FOR z/OS - DBHDA                "HD ANALYSIS REPORT"                                PAGE:     2
5655-K53                                                  DATE: 07/10/2006  TIME: 17.37.25                  FABGHIST - V2.R2

                                                                                    SPECIFIED DATA        PREVIOUS DATA

                                                                                DATE: 07/10/2006      DATE:  NONE
HD TUNING STATISTICS                                                            TIME:   17:33:53      TIME:  NONE
--------------------                                                    (CREATED BY: HDPC)    (CREATED BY: NONE)
  DIRECT ALGORITHM NAME                                 -                         DFSHDC40
  LONGEST SEGMENT IN DATA SET                           -                              246
  HIGH BLOCK NUMBER IN RAA                              -                             4500
  RAPS PER BLOCK                                        -                                1
  TOTAL RAPS                                            -                             4500
  BYTE LIMIT COUNT                                      -                              N/A
  AVG. DATABASE RECORD LENGTH                           -                              793
  FREE SPACE SCAN CYLINDERS                             -                                0
  FSPC BLK. EVERY N BLKS                                -                                0
  % FSPC WITHIN EACH BLK                                -                                0
  NO. KEY RECORDS WRITTEN                               -                                0

  ROOTS IN HOME BLOCK                                   -                     2991    27 %
  ROOTS 1 BLOCK AWAY                                    -                       35     0 %
  ROOTS BEYOND                                          -                     1244    11 %
  ROOTS IN OVERFLOW                                     -                     6730    61 %

  BLOCKS WITHOUT ROOTS IN RAA                           -                     1436    31 %
  AVG. COUNT OF ROOTS PER ACT. BLK IN RAA               -                                1.3
  AVG. COUNT OF ROOTS PER ACTIVE RAP                    -                                2.6
  COUNT OF RAPS NOT USED                                -                      410     9 %
```

When should you reorganize

1. Database performance has deteriorated.
2. There are too many physical I/Os to DASD.
3. The database structure has changed.
   - For example, you should reorganize a HALDB partition after changing its boundaries or high key.
   - The (P)HDAM randomizer has changed.
   - The HALDB Partitions Selection exit routine has changed.
4. When the OSAM or VSAM data set goes into extents.
5. When the data portion of a VSAM data set High-Used RBA keeps increasing.
6. When the index portion of a VSAM data set keeps having CI and CA splits.
7. When you start to run out of free space in the database.
8. When roots start not to randomize to the home block in a (P)HDAM database, and start to go to the beyond area or to overflow

Reorganizations

Reorganization Process

- Use of Standard IMS Utilities
- Other Options
  - Faster utilities
  - Read-Only Reorg
  - High Availability Reorgs
  - Zero Outage Reorgs
  - Conversion Reorgs – moving to HALDB

Reducing Frequency of Reorgs

Know your applications

Growth Pattern

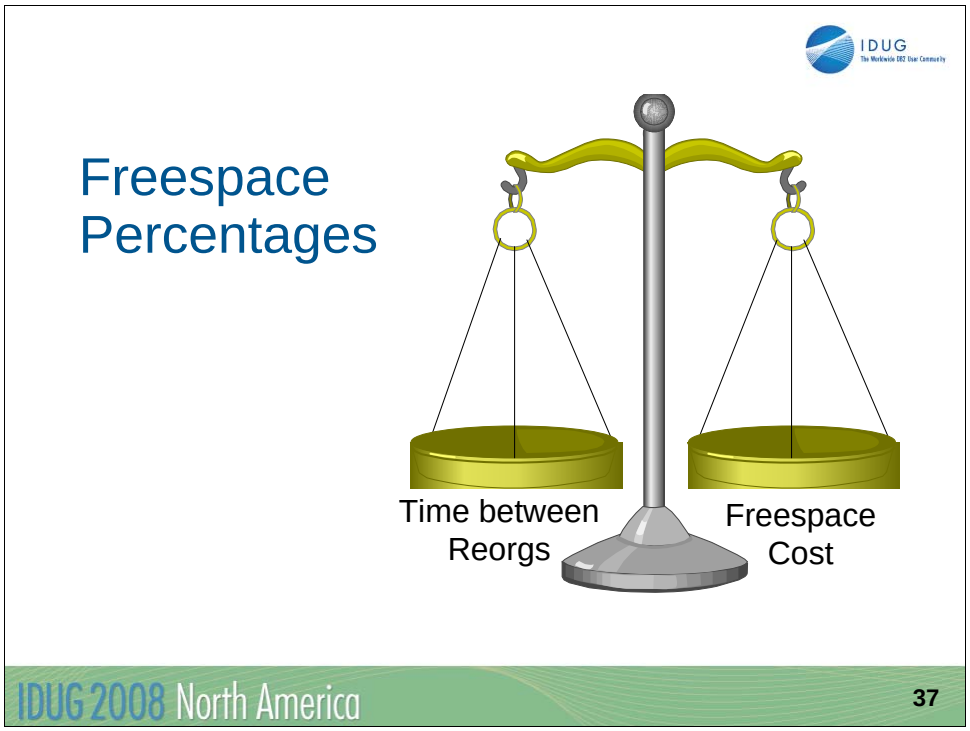
- What is the key based on?
  - Customer account number?
  - State
  - Date
- Growth
  - Random
  - At the end?

What else?

- Regular transactions
  - Types of updates
  - Inserts?
  - Updates
  - Delete
    - Regular archival process?
- Where are the updates
  - Which segments
  - One segment growth?
- Regular Cyclic activity
  - Massive insert
  - Massive delete

Read Pattern

- Random
- Sequential
- Individual Segments
- Large Sequential areas or entire database
- Most frequent transactions
- Most critical transactions



More freespace means data is most-likely going to be where it belongs

Random Access will have less I/Os

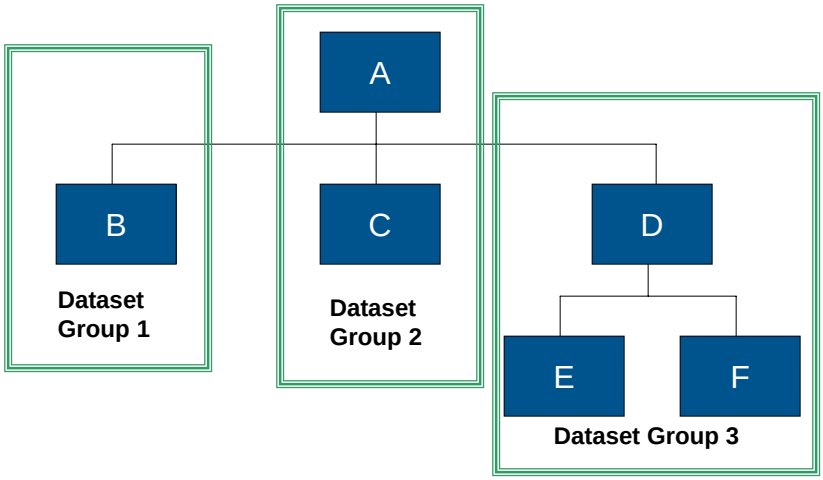
Growth Patterns & Freespace

Reduce number of reorgs by getting this right

- No additions:
  - No need for FREESPACE.
- Few additions:
  - No FREESPACE or some FREESPACE in the CI.
- Evenly distributed additions:
  - FREESPACE in the CA or FREESPACE in both CI and CA.
- Unevenly distributed additions:
  - Specify a small amount of FREESPACE.
- Additions all at the end
  - No FREESPACE – EXTRA SPACE INSTEAD

IMS HP Pointer Checker
IMS Performance Analyzer

Multiple Dataset Groups



Reorganization Performance Opportunities

Database Performance

- **Access Methods**
- **Block sizes, CI sizes and Record sizes**
- **Free Space**
- Randomization Parameters
- Fixed Length vs. Variable Length
- **Pointer Options**
- SCAN parameter on the DATASET statement
- **Multiple data set groups**
- **Compression**
- Encryption
- Secondary Indexes
- Fast Path considerations
- Non-Recoverable databases
- **OSAM vs. VSAM**
- **Buffer Life Concept**

Database Access Methods – performance

To choose an IMS access method:

- What type of processing is done (Choices are shown in preferred order)?
  - Direct: Use DEDB, HDAM, HIDAM, or HISAM.
  - Sequential: Use DEDB (Seq Rand), HDAM (Seq Rand), HIDAM, or HISAM.
  - Both: Use DEDB (Seq Rand), HDAM (Seq RAND), or HIDAM.
- Is the data volatile? Yes, use DEDB, HDAM, or HIDAM.
- Do the database records vary in length? Yes, use DEDB, HDAM, or HIDAM.
- Are logical relationships needed? Yes, use HDAM or HIDAM.
- Are secondary indexes needed? Yes, use HDAM or HIDAM.
- Is there a need for a journaling capability? Yes, use DEDB.

Note: Wherever HDAM or HIDAM is shown, partitioning (HALDB) is preferred. Seq Rand means using a Randomizer that maintains the key sequence.

Block or CI Size

Larger CIs or blocks:

- Improve sequential processing.
- Reduce the number of IWAITS.
- Increase IWAIT time per IWAIT.
- Decrease total IWAIT time.

Smaller CIs or blocks might:

- Improve random processing.
- Increase number of IWAITS.
- Reduce IWAIT time per IWAIT.

IMS Performance Analyzer

Pointer TidBits

- Use child and twin pointers instead of hierarchic pointers.
- Do not specify twin backward pointers for dependent segments unless you satisfy the criteria for deletes with logical relationships.
- Never specify twin forward only pointers for HIDAM roots.
- Never specify twin forward and backward pointers for HDAM roots.
- Specify no twin pointers for HIDAM and PHIDAM roots.
- If you specify RULES=(,LAST) or use last as the default for segments without sequence fields, you should define a physical child last pointer from the parent if there might be a long twin chain.

Compression Tidbits

The considerations are:

- Improves DASD space utilization (more data in block)
- Improves buffer space utilization
- Might reduce I/O
- Increases CPU time unless you are using Hardware Data Compression

IMS HP Pointer Checker
IMS Hardware Compression Ext

Database – OSAM vs VSAM

- Tests were run in a controlled environment in the Silicon Valley Laboratory using 10 HIDAM databases.
- The first set of tests were run with the databases defined with VSAM, and then a second set of tests were run with OSAM using the same workload that was used in the first test.
- Set one
  - three BMPs each executing 2 000 000 total database calls.
  - There were 10 qualified GHU calls performed along with 1 000 000 qualified GHN calls and 1 000 000 replace calls.
- Set two
  - four BMPs each executing 4 500 000 total database calls.
  - There was one qualified GHU call performed along with 1 000 qualified GHN calls, 1 000 replace calls, and 4 000 000 GN calls

Database – OSAM vs VSAM

Type	Avg CPU Time	Elapsed Time	Delta CPU	Delta Elapsed
BMP Set One				
VSAM	168	8.71		
OSAM	136	6.01	19.04% reduction	27.59 % reduction
OSAM SB	138	6.93	18.8% reduction	27.34% reduction

BMP Set Two				
VSAM	98	5.45		
OSAM	57	3.50	41.83% reduction	35.78% reduction
OSAM SB	61	1.16	37.75% reduction	78.59% reduction

OSAM vs VSAM ---- Why??

- OSAM writing of multiple blocks
  - Sorts by physical location
  - Chained writes in parallel
- Shorter processor instruction path length
- OSAM sequential buffering
- OSAM data sets up to 8 Gb
- Reuse OSAM data sets

What other type of Performance Tuning to consider

- Bufferpool Tuning
  - Most important statistic is Buffer Life
  - Changing buffer requires taking IMS Down
    - Need to be able to predict result so multiple outages to correct changes not needed
  - Consider moving most active DBs to their own subpool
    - Run predictive reports before attempting change
  - Often find may subpools can be reduced or removed – freeing resources
  - Dramatic performance improvements possible

IMS Performance Analyzer
IMS Buffer Pool Analyzer

Where to go for more IMS Hints

Redbooks (www.redbooks.ibm.com):

- IMS Performance and Tuning Guide
  - SG24-7324-00
- IMS Primer
  - SG24-5352-00

Reference Book

- **An Introduction to IMS: Your Complete Guide to IBM's Information Management System**
  - Available from www.amazon.com

Reorganization in Summary

- Prime criteria should be application performance
- Exception performance reports
- Exception database statistic reports
- Use Reorganizations as performance tuning opportunities
- Publish your successes

IBM Performance Tools

- IMS HP Pointer Checker
- IMS Performance Analyzer
- IMS Buffer Pool Analyzer
- IMS Connect Extensions

IBM Reorganization Tools

- IMS Parallel Reorganization
- IMS Online Reorganization Facility
- IMS HP Unload
- IMS HP Load
- IMS Index Builder
- IMS HP Prefix Resolution

Contact me

- Janet LeBlanc
  - leblancj@ca.ibm.com

IDUG
The Worldwide DB2 User Community

Q & A